# On the Performance of Neural Networks for Face Recognition: Linear or Nonlinear Classifiers?

Rafael O. Lima and Guilherme A. Barreto

Department of Teleinformatics Engineering, Federal University of Ceará
Av. Mister Hull, S/N - Campus of Pici, Fortaleza, Ceará, Brazil
professor.rlima@gmail.com,guilherme@deti.ufc.br

**Abstract.** The main goal of this study is to empirically evaluate linear and nonlinear neural network based classifiers to be embedded in mobile devices. For this purpose, this paper reports a comprehensive performance comparison study involving 10 neural network models for human face recognition. All the classifiers are evaluated on two benchmarking face databases. For the linear classifiers we evaluate eight variants of the LMS and perceptron learning rules, while for the nonlinear ones we evaluate two state-of-the-art classifiers. In addition, we also evaluate empirically the robustness of all classifiers to the presence of gaussian and impulsive noise in test images. The results of the experiments indicate that the linear classifiers perform as good as nonlinear ones, with the advantage of demanding much lower computational resources.

**Keywords:** Face recognition, neural networks, linear classifiers, nonlinear classifiers, embedded applications

## 1 Introduction

For the last two and a half decades the field of human face recognition has been experiencing remarkable advances in terms of performance, so that it has been drawing wide interest among both research and commercial communities alike. These advances, for instance, have turned face recognition into an important component in several commercially available products for biometrics [1, 18].

Though biometric applications that verify a person's identity based on their physical attributes, such as fingerprints, face, voice or iris, have been in use for some time, only with the recent developments of 3G and 3.5G cellular radio technology, biometrics has become feasible to be embedded in mobile devices for entertainment or security purposes [10, 15, 14]. Such applications have different constraints in terms of complexity of processing requirements and thus present several performance challenges for machine learning classification methods.

In particular, for face recognition most of the applications involves the implementation of machine learning classifiers, such as multilayer perceptron, radial basis functions, support vector machines, self-organizing maps, gaussian process and $K$-nearest neighbors classifiers, among several others. However, usually the

classifiers are evaluated off-line without considering the cost of its implementation, an issue that becomes critical in real-time embedded applications.

From the exposed, the goal of this paper is to present a systematic performance comparison of 10 neural network classifiers, linear and nonlinear ones, on face recognition tasks. All the algorithms are evaluated on a benchmarking face database, which contain photographs of individuals under different poses, configurations and facial expressions. We also aim at evaluating the robustness of the classifers to gaussian and impulsive noise. Besides the recognition rate, the performance of the classifiers are discussed taking into consideration the number of parameters to be stored.

The remainder of the paper is organized as follows. In Sections 2 and 3 we briefly describe all the classifiers to be evaluated in this paper. In Section 4 the performance comparison of all classifiers is carried out and the results are discussed. The paper is concluded in Section 5.

## 2 Linear Classifiers

For all classifiers to be described, we have used the following decision criterion:

$$\hat{\omega}(t) = \omega_k(t), \quad \text{where } k = \arg \max_{i=1,\ldots,c} \{y_i(t)\}, \tag{1}$$

where $c$ is the number of classes. Furthermore, the output label vector $\mathbf{d}$, used as desired response during the training of these classifiers, are represented with only one of its components set to "+1", while the others are set to "-1". All the adjustable parameters are initialized randomly within the range $[-0.5, +0.5]$.

**Simple Perceptron** [11]: The output and the learning rule of the $i$-th neuron of the simple perceptron (SP) classifier are given by:

$$y_i(t) = \text{sign}\left(\sum_{j=0}^{p} w_{ij}(t)x_j(t)\right) = \text{sign}\left(\mathbf{w}_i^T(t)\mathbf{x}(t)\right), \tag{2}$$

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e_i(t)\mathbf{x}(t), \tag{3}$$

where $\text{sign}(\cdot)$ is the sign function, $x_j(t)$ is the $j$-th input feature, $w_{ij}(t)$ is the weight connecting $j$-th input to the $i$-th output neuron, $w_{i0} = b_i$ is the bias of the $i$-th output neuron, $x_0(t) = -1$, $0 < \eta < 1$ is the learning rate and $e_i(t) = d_i(t) - y_i(t)$ is the error of the $i$-th neuron, for a given desired response $d_i(t)$. For simplicity of notation, all the inputs $x_0(t), x_1(t), ..., x_p(t)$ are grouped into an input vector $\mathbf{x}(t) \in \mathbb{R}^{p+1}$ and the weights and bias $w_{i0}(t), w_{i1}(t), ..., w_{ip}(t)$ of the $i$-th output neuron are grouped into a weight vector $\mathbf{w}_i(t) \in \mathbb{R}^{p+1}$. The superscript $T$ denotes the transpose of a vector and $t$ denotes the iteration.

**Logistic Perceptron** [13]: The logistic perceptron (LP) is simply a perceptron with a sigmoidal output. Thus, the output and the learning rule of the $i$-th neuron of this classifier are given by

$$y_i(t) = \tanh\left(\mathbf{w}_i^T(t)\mathbf{x}(t)\right) \quad \text{and} \quad \mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e_i(t)\left(1 - y_i^2(t)\right)\mathbf{x}(t), \tag{4}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function. Note that the learning rule is slightly altered by the inclusion of the factor $1 - y_i^2(t)$ (derivative of the output at time $t$). This factor slows learning if the output is close to the saturation regions (i.e. $y_i(t) \approx \pm 1$).

**Madaline/LMS** [16]: The Madaline classifier is a network of linear neurons, with the same architecture of the SP network. The main difference between SP and Madaline networks is in the output activation function. The SP network uses the `sign` function, while the Madaline uses the identity function. The consequence is that the output value $y_i(t)$ in Madaline networks is a real number, while in the SP network it assume only two values, since it is quantized to $\{-1, +1\}$. In a nutshell, the output and the learning rule of the $i$-th neuron of the Madaline/LMS classifier are given by

$$y_i(t) = \mathbf{w}_i^T(t)\mathbf{x}(t) \quad \text{and} \quad \mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e_i(t)\mathbf{x}(t). \tag{5}$$

The SP learning rule and the LMS learning rule are essentially the same equation, however, the error term $e_i(t)$ in Equation (2) is a discrete variable, while in Equation (5) it is a continuous variable, since there is no nonlinear (i.e. `sign`)function at the output of Madaline's neurons.

**Madaline/normalized-LMS** [4]: The output and the learning rule of the $i$-th neuron of the Madaline classifier, trained with the normalized LMS (NLMS) rule, are given by

$$y_i(t) = \mathbf{w}_i^T(t)\mathbf{x}(t) \quad \text{and} \quad \mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \frac{\eta}{\alpha + \|\mathbf{x}(t)\|^2} e_i(t)\mathbf{x}(t), \tag{6}$$

where $\alpha \ll 1$ is a very small constant used to avoid division by zero. The inclusion of the normalization term $\|\mathbf{x}(t)\|^2$ aims at making the algorithm convergence more independent of the energy (or power) of the input signal.

**Madaline/sign-LMS** [4]: The output and the learning rule of the $i$-th neuron of the Madaline classifier, trained with the sign-LMS (SLMS) rule, are given by

$$y_i(t) = \mathbf{w}_i^T(t)\mathbf{x}(t) \quad \text{and} \quad \mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta \text{sign}\left(e_i(t)\right)\mathbf{x}(t). \tag{7}$$

Note that in the SLMS rule only the sign of the error term is taken into account in updating the weight vectors (the error magnitude is disconsidered). The consequence is that it is computationally faster than the LMS algorithm (no multiplication between $e_i$ and $\mathbf{x}$ is required), but it converges much slower.

**Madaline/median-LMS** [17]: The output and the learning rule of the $i$-th neuron of the Madaline classifier, trained with the median-LMS (MLMS) rule, are given by:

$$\begin{aligned} y_i(t) &= \mathbf{w}_i^T(t)\mathbf{x}(t), \\ \mathbf{w}_i(t+1) &= \mathbf{w}_i(t) + \eta \text{med}\left[e_i(t)\mathbf{x}(t), e_i(t-1)\mathbf{x}(t-1), ..., e_i(t-\tau)\mathbf{x}(t-\tau)\right], \end{aligned} \tag{8}$$

where $\text{med}(\cdot)$ is the median operator and $\tau > 0$ is an integer. The MLMS rule provides a smoothing effect that may be useful when impulsive noise is present.

If noise is not impulsive, the performance of median-LMS are comparable with those of LMS, thus the extra computational cost of MLMS is not worth.

**Madaline/Leaky-LMS** [8]: The output and the learning rule of the $i$-th neuron of the Madaline classifier are given by:

$$y_i(t) = \mathbf{w}_i^T(t)\mathbf{x}(t), \tag{9}$$

$$\mathbf{w}_i(t+1) = (1-\lambda)\mathbf{w}_i(t) + \eta e_i(t)\mathbf{x}(t). \tag{10}$$

where $0 < \lambda < 1$ is a decaying parameter. The inclusion of the factor $(1-\lambda)$ is equivalent to add gaussian white noise with zero mean and variance $\lambda/\eta$ to the input vector $\mathbf{x}(t)$ [4]. This property may increase the robustness of the Madaline classifier to gaussian noise.

**Optimal Linear Associative Memory (OLAM)** [7]: The output and the learning rule of the $i$-th neuron of the OLAM classifier are given by

$$y_i(t) = \mathbf{w}_i^T(t)\mathbf{x}(t) \quad \text{and} \quad \mathbf{W} = \mathbf{D}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T, \tag{11}$$

where $\mathbf{X} = [\mathbf{x}_1|\mathbf{x}_2|\ldots|\mathbf{x}_N]$ is the input data matrix ($\mathbf{X} \in \mathbb{R}^{p\times N}$) and $\mathbf{D} = [\mathbf{d}_1|\mathbf{d}_2|\ldots|\mathbf{d}_N]$ is the corresponding matrix of desired output vectors ($\mathbf{D} \in \mathbb{R}^{c\times N}$). The weight vector $\mathbf{w}_i$ corresponds to the $i$-th row of $\mathbf{W}$. The OLAM network can be seen as a Madaline network, whose weights are computed through the (nonrecursive) least squares method.

## 3 Nonlinear Classifiers

**Multilayer Perceptron Classifier (MLP)** [3]: The MLP network consists of an input with $p$ features, a hidden layer with $q$ neurons with sigmoidal activation functions and an output layer with $c$ neurons (also with sigmoidal activation functions). The output of the $k$-th neuron $y_i(t)$, $k = 1, \ldots, c$, is given by

$$y_k(t) = \tanh\left[\sum_{i=1}^{q} m_{ki}(t)\tanh\left(\sum_{j=0}^{p} w_{ij}(t)x_j(t)\right) - \theta_k(t)\right], \tag{12}$$

where $\mathbf{x}(t)$ is the current input vector feature, $\mathbf{w}_i(t) = [w_{i1} \;\; w_{i2} \;\; \cdots \;\; w_{iq}]^T$ is the weight vector of the $i$-th hidden neuron, $m_{ki}$ is the weight connecting $i$-th hidden neuron to the $k$-th output neuron. The parameter $\theta_k$ is the bias of the $k$-th output neuron. The weights and biases are computed through the standard backpropagation algorithm.

**Extreme Learning Machine (ELM)** [6]: The ELM network is a recent development in the field of neural computation. Its main idea is to combine the nonlinear pattern recognition power of multilayer feedforward networks with a fast training algorithm. In terms of topology and parameters, ELM is exactly equal to the one-hidden-layered MLP network, with linear output neurons.

In the first stage of the training algorithm, the weights $w_{ij}$ are randomly initialized and kept unchanged. In the second stage, the training vector are presented one-by-one and the activations of the hidden neurons are stored as columns of a matrix $X$ (the number of columns of $X$ is equal to the number of training patters, while the number of rows is equal to the number of hidden neurons.). Finally, in the third stage, the weights $m_{ki}$ are computed by the standard least squares method. Due to its fast training, the ELM network is particularly suitable for embedded applications.

It is worth mentioning that, for embedded applications, even with today's hardware technology for mobile devices, memory space is still a limited, costly resource [9]. Thus, when choosing between two classifiers of similar recognition rates, the lesser the number of parameters to be stored, the better is the classifier.

All the linear classifers described in Subsection 2 requires the storage of $p+1$ parameters ($p$ weights and one bias). The MLP and ELM networks, each, require the storage of $(p+1)\cdot q+(q+1)\cdot c$ parameters, which is a much higher requirement than that of linear neural classifiers. Hence, between a linear and a nonlinear neural classifier with similar recognition rates, one should choose the linear one. Details about the time complexity of the adaptive filtering algorithms used by the classifiers described previously can be found in [12].

## 4 Computer Experiments

The Yale-B face image database [2] contains 5760 single light source images of 10 subjects each seen under 576 viewing conditions (9 poses x 64 illumination conditions). For every subject in a particular pose, an image with ambient (background) illumination was also captured. Hence, the total number of images is in fact $5760 + 90 = 5850$. The size of each image is $640 \times 480$. The Sussex Image Database [5] contains 100 images of $384 \times 287$ pixels. These images are in greyscale in the *Sun Standard Rasterfile* format. This database has 10 subjects. Each subjects is seen under 10 different poses. Pixel intensities of both image databases are rescaled to the range $[-1, 1]$. Each original image in the Yale-B and Sussex databases are firstly reduced to $50 \times 50$ pixels, then its columns are rearranged into a single column-vector of dimension $p = 2,500$. Principal component analysis (PCA) is then performed in order to project the resulting 2500-dimensional vectors onto the first 35 principal directions, explaining 95% of the variance of the original data.

The results are shown in Tables 1 and 2. The statistics shown in the tables were collected over 50 training/testing realizations. For each realization, the training and testing data vectors are randomly selected in a proportion of 80% for training and 20% for testing. The adjustable parameters are randomly initialized for each training/testing realizations. For all the classifiers (except OLAM and ELM) the learning rate was set to $\eta = 0.01$. For the MLP and ELM classifiers the number of hidden neurons was set to 80 and 62 respectively. For all neural classifiers (except OLAM and ELM) the number of training epochs was set to 60. For the NLMS and LLMS rules, we set $\alpha = 0.001$ and $\lambda = 0.01$, respectively.

**Table 1.** Performances for the Yale-B database.

| **Neural Models** | **Recognition rates (%)** | | | |
|---|---|---|---|---|
| | $min$ | $mean$ | $max$ | $std-dev$ |
| SP | 96.8393 | 98.4872 | 100 | 0.1167 |
| **LP** | **99.9573** | **99.9987** | **100** | **0.0067** |
| OLAM | 98.48 | 98.79 | 100 | 0.0192 |
| LMS | 99.7650 | 99.9158 | 99.9786 | 0.0531 |
| N-LMS | 99.5726 | 99.8940 | 100 | 0.0939 |
| S-LMS | 65.2137 | 72.8051 | 79.7436 | 3.0921 |
| **M-LMS** | **99.7436** | **99.9812** | **100** | **0.0466** |
| **L-LMS** | **99.4872** | **99.8923** | **100** | **0.1169** |
| MLP | 99.7899 | 99.8778 | 100 | 1.3121 |
| ELM | 93.5043 | 96.4803 | 98.4615 | 1.3124 |

**Table 2.** Performances for the Sussex database.

| **Neural Models** | **Recognition rates (%)** | | | |
|---|---|---|---|---|
| | $min$ | $mean$ | $max$ | $std-dev$ |
| SP | 55 | 82.4 | 100 | 10.3628 |
| LP | 90 | 98.2 | 100 | 3.1558 |
| **OLAM** | **90** | **97.8** | **100** | **2.6361** |
| LMS | 90 | 97.8 | 100 | 3.2198 |
| **N-LMS** | **90** | **98.5** | **100** | **2.5254** |
| S-LMS | 80 | 94.3 | 100 | 4.9497 |
| M-LMS | 85 | 95.3 | 100 | 4.3342 |
| L-LMS | 90 | 98 | 100 | 3.1944 |
| MLP | 85 | 97.8 | 100 | 3.5225 |
| **ELM** | **90** | **98.6** | **100** | **2.4826** |

All the experiments were simulated in the Matlab environment in a MacBookPro with 2.26 GHz and 4GB ram, running MacOS X 10.6 operating system.

The results for the three best classifiers are highlighted in the tables. Several interesting conclusions can be drawn from the results. For the YALE-B database, several linear classifiers achieved average performances equivalent to the best nonlinear classifier (i.e. the MLP). The standard deviation of the performance of the linear classifiers, which are much lower that that of the MLP classifier. For the Sussex database, the same pattern was observed, that is, the linear classifier again achieved average performances equivalent to the best nonlinear classifier (the ELM, this time). The standard deviation of the results were also equivalent. For these experiments, considering a trade-off between the number of parameters, time complexity and the classification performance, we recommend the LP and OLAM classifiers for embedded applications in mobile devices.

We have further evaluated the all classifiers by adding Gaussian and impulsive (salt and pepper) noise to the test images, with varying intensity. In the experiments, the variance of the gaussian noise and the density of the impul-

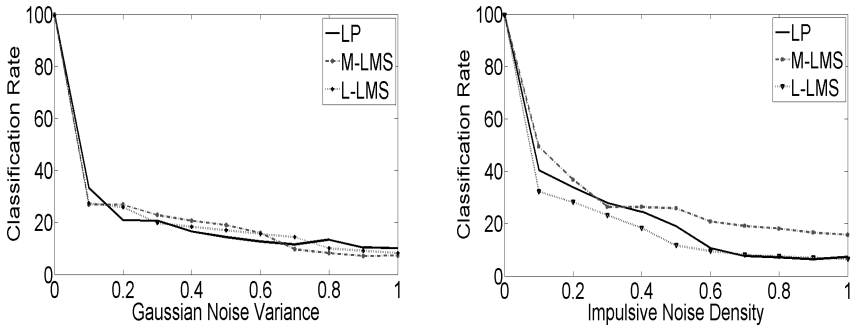


**Fig. 1.** Performances of linear and nonlinear neural classifiers to test images (YALE-B) corrupted by (a) gaussian and (b) impulsive noise.

sive noise were varied from 0 to 1, in increments of 0.1. For each value of the variance/density, the classifiers were trained with noise free images under the same methodology described in the first experiments. The results are shown in Figures 1 and 2 for the three best classifiers in each database. From the figures, one can infer that the average performances of the two linear classifers (LP and LLMS) and the two nonlinear classifiers under the presence of gaussian/impulsive noise are equivalent, following the pattern observed in the noise-free experiments.

## 5 Conclusions

In this paper we reported a performance comparison study of 10 neural network models for human face recognition. We evaluated eight variants of the Perceptron and LMS learning rules, and two state-of-the-art nonlinear classifiers. We also evaluate empirically the robustness of all classifiers to the presence of gaussian and impulsive noise in test images. The results of the experiments indicate that the linear classifiers can perform as good as nonlinear ones for the task of interest, with the advantage of demanding much lower computational resources.

## References

1. Dorizzi, B.: New trends in biometrics. In: Telecommunications: Advances and Trends in Transmission, Networking and Applications, pp. 157–171. Edson Queiroz Foundation, Ceará, Brazil (2006)
2. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(6), 643–660 (2001)
3. Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan Publishing Company, Englewood Cliffs, NJ (1994)

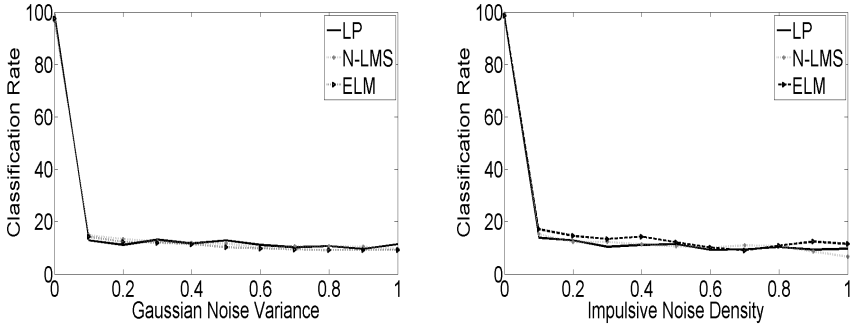


**Fig. 2.** Performances of linear and nonlinear neural classifiers to test images (SUSSEX) corrupted by (a) gaussian and (b) impulsive noise.

4. Haykin, S.: Adaptive Filter Theory. Prentice-Hall, 4th edn. (2001)
5. Howell, A.J.: Automatic Face Recognition using Radial Basis Function Networks. Master's thesis, University of Sussex, Brighton, UK (September 1997)
6. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing 70(1–3), 489–501 (2006)
7. Kohonen, T., Ruohonen, M.: Representation of associated data by matrix operators. IEEE Transactions on Computers 22(7), 701–702 (1973)
8. Mayyas, K., Aboulnasr, T.: Leaky LMS algorithm: MSE analysis for gaussian data. IEEE Transactions on Signal Processing 45(4), 927–934 (1997)
9. Novak, M.: Algorithm optimizations: Low computational complexity. In: Tan, Z.H., Lindberg, B. (eds.) Automatic Speech Recognition on Mobile Devices and Over Communication Networks, pp. 213–231. Springer (2008)
10. Pocovnicu, A.: Biometric security for cell phones. Informatica Economica 13(1), 57–63 (2009)
11. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408 (1958)
12. Sayed, A.H.: Adaptive Filters. Wiley-IEEE Press (2008)
13. Shynk, J.J.: Performance surfaces of a single-layer Perceptron. IEEE Transactions on Neural Networks 1(3), 268–274 (1990)
14. Varga, I., Kiss, I.: Speech recognition in mobile phones. In: Tan, Z.H., Lindberg, B. (eds.) Automatic Speech Recognition on Mobile Devices and Over Communication Networks, pp. 301–325. Springer (2008)
15. Wang, J.F., Wang, J.C., Mo, M.H., Tu, C.I., Lin, S.C.: The design of a speech interactivity embedded module and its applications for mobile consumer devices. IEEE Transactions on Consumer Eletronics 54(2), 870–876 (2008)
16. Widrow, B.: Thinking about thinking: The discovery of the LMS algorithm. IEEE Signal Processing Magazine 22(1), 100–106 (2005)
17. Williamson, G.A., Clarkson, P.M., Sethares, W.A.: Performance characteristics of the median LMS adaptive filter. IEEE Transactions on Signal Processing 41(2), 667–680 (1993)
18. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Computing Surveys 35(4), 399–458 (2003)